Trilinos Overview

Michael A. Heroux
Sandia National Laboratories
with contributions from many collaborators

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Why Use Mathematical Libraries?

- A farmer had chickens and pigs. There was a total of 60 heads and 200 feet. How many chickens and how many chickens did the farmer have?
- Let x be the number of chickens, y be the number of pigs.
- Then:

$$\begin{aligned} x + y &= 60 \\ 2x + 4y &= 200 \end{aligned}$$

- From first equation $x = 60 - y$, so replace x in second equation:

$$2(60 - y) + 4y = 200$$

- Solve for y:

$$\begin{aligned} 120 - 2y + 4y &= 200 \\ 2y &= 80 \\ y &= 40 \end{aligned}$$

- Solve for x: $x = 60 - 40 = 20$.
- The farmer has 20 chickens and 40 pigs.

- A restaurant owner purchased one box of frozen chicken and another box of frozen pork for \$60. Later the owner purchased 2 boxes of chicken and 4 boxes of pork for \$200. What is the cost of a box of frozen chicken and a box of frozen pork?
- Let x be the price of a box of chicken, y the price of a box of pork.
- Then:

$$\begin{aligned} x + y &= 60 \\ 2x + 4y &= 200 \end{aligned}$$

- From first equation $x = 60 - y$, so replace x in second equation:

$$2(60 - y) + 4y = 200$$

- Solve for y:

$$\begin{aligned} 120 - 2y + 4y &= 200 \\ 2y &= 80 \\ y &= 40 \end{aligned}$$

- Solve for x: $x = 60 - 40 = 20$.
- A box of chicken costs \$20 and a box of pork costs \$40.

Problem Statement

- A restaurant owner purchased one box of frozen chicken and another box of frozen pork for \$60. Later the owner purchased 2 boxes of chicken and 4 boxes of pork for \$200. What is the cost of a box of frozen chicken and a box of frozen pork?

Math Translation

- Let x be the price of a box of chicken, y the price of a box of pork.

Problem Setup

- Then:

$$x + y = 60$$
$$2x + 4y = 200$$

Solution Method

- From first equation $x = 60 - y$, so replace x in second equation:

$$2(60 - y) + 4y = 200$$

- Solve for y:

$$120 - 2y + 4y = 200$$
$$2y = 80$$
$$y = 40$$

- Solve for x: $x = 60 - 40 = 20$.

Translate Back

- A box of chicken costs \$20. A box of pork costs \$40.

Why Math Libraries?

- Many types of problems.
- Similar Mathematics.
- Separation of concerns:
  - Problem Statement.
  - Translation to Math.
  - Set up problem.
  - Solve Problem.
  - Translate Back.

Sandia National Laboratories

Importance of Math Libraries

- Computer solution of math problems is hard:
  - Floating point arithmetic not exact:
    - $1 + \varepsilon = 1$, for small $\varepsilon > 0$.
    - $(a + b) + c$ not always equal to $a + (b + c)$.
  - Hi fidelity leads to large problems: 1M to 10B equations.
  - Clusters require coordinated solution across 100 – 1M processors.
- Sophisticated solution algorithms and libraries leveraged:
  - Solver expertise highly specialized, expensive.
  - Write code once, use in many settings.
- Trilinos is a large collection of state-of-the-art work:
  - The latest in scientific algorithms.
  - Leading edge software design and architecture.

Background/Motivation

Trilinos

- R&D 100 Winner
- 9000 Registered Users.
- 30,000 Downloads.
- Open Source.

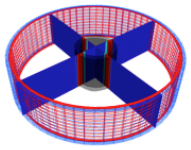
Laptops to Leadership systems

Optimal Kernels to Optimal Solutions:

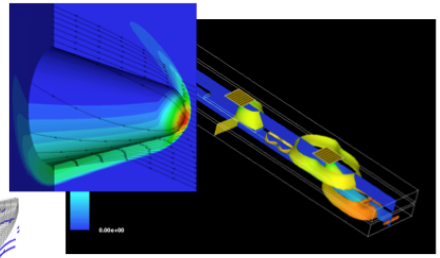
- Geometry, Meshing
- Discretizations, Load Balancing.
- Scalable Linear, Nonlinear, Eigen, Transient, Optimization, UQ solvers.
- Scalable I/O, GPU, Manycore

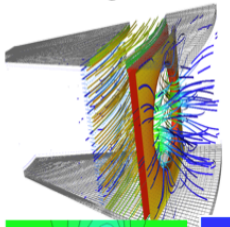
- 60 Packages.
- Binary distributions:
  - Cray LIBSCI
  - Debian, Ubuntu
  - Intel (in process)

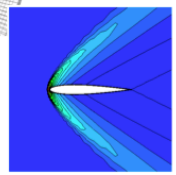
Transforming Computational Analysis To Support High Consequence Decisions



PDES

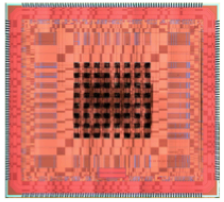




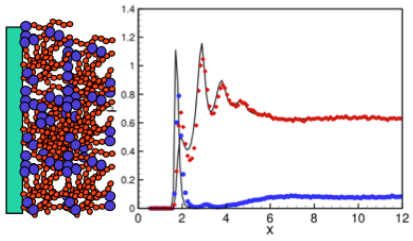


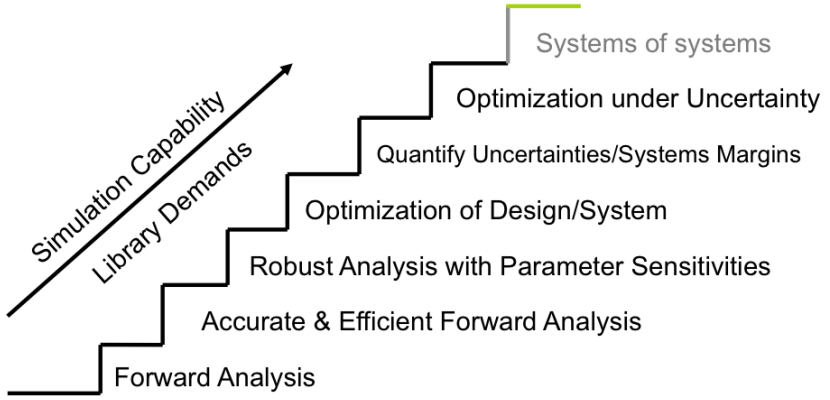
And More…

Circuits



Inhomogeneous Fluids





Each stage requires *greater performance* and *error control* of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.

Applications

- All kinds of physical simulations:
  - Structural mechanics (statics and dynamics)
  - Circuit simulations (physical models)
  - Electromagnetics, plasmas, and superconductors
  - Combustion and fluid flow (at macro- and nanoscales)
- Coupled / multiphysics models
- Data and graph analysis (2D distributions).

Trilinos Strategic Goals

- Scalable Computations: As problem size and processor counts increase, the cost of the computation will remain nearly fixed.
- Hardened Computations: Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- Full Vertical Coverage: Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

Algorithmic Goals

- *Grand* Universal Interoperability: All Trilinos packages, and important external packages, will be interoperable, so that any combination of packages and external software (e.g., PETSc, Hypre) that makes sense algorithmically will be possible within Trilinos.
- Universal Accessibility: All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.
- TriBITS Lifecycle: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: "Self-sustaining".

Software Goals

Capability Leaders: Layer of Proactive Leadership

- Areas:
  - **User Experience (W. Spotz) (May 2012).**
  - Scalable I/O: (R. Oldfield) (Nov 2010).
  - Framework, Tools & Interfaces (J. Willenbring).
  - Software Engineering Technologies and Integration (R. Bartlett).
  - Discretizations (P. Bochev).
  - Geometry, Meshing & Load Balancing (K. Devine).
  - Scalable Linear Algebra (M. Heroux).
  - Linear & Eigen Solvers (J. Hu).
  - Nonlinear, Transient & Optimization Solvers (A. Salinger).
- Each leader provides strategic direction across all Trilinos packages within area.

Unique features of Trilinos

- Huge library of algorithms
  - Linear and nonlinear solvers, preconditioners, …
  - Optimization, transients, sensitivities, uncertainty, …
- Growing support for multicore & hybrid CPU/GPU
  - Built into the new Tpetra linear algebra objects
    - Therefore into iterative solvers with zero effort!
  - Unified intranode programming model
  - Spreading into the whole stack:
    - Multigrid, sparse factorizations, element assembly…
- Growing support for mixed and arbitrary precisions
  - Don’t have to rebuild Trilinos to use it!
- Growing support for huge (> 2B unknowns) problems

Trilinos' software organization

Sandia National Laboratories

Trilinos is made of packages

- Not a monolithic piece of software
  - Like LEGO™ bricks, not Matlab™
- Each package:
  - Has its own development team and management
  - Makes its own decisions about algorithms, coding style, etc.
  - May or may not depend on other Trilinos packages
- Trilinos is not “indivisible”
  - You don’t need all of Trilinos to get things done
  - Any subset of packages can be combined and distributed
  - Current public release (11.2) contains 54 of the 60+ Trilinos packages
- Trilinos top layer framework
  - Not a large amount of source code: ~1.5%
  - Manages package dependencies
    - Like a GNU/Linux package manager
  - Runs packages’ tests nightly, and on every check-in
- Package model supports multifrontal development

Trilinos Package Summary

	Objective	Package(s)
Discretizations	Meshing & Discretizations	STK, Intrepid, Pamgen, Sundance, ITAPS, Mesquite
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Jpetra, Tpetra, Kokkos
	Interfaces	Thyra, Stratimikos, RTOp, FEI, Shards
	Load Balancing	Zoltan, Isorropia, Zoltan2
	“Skins”	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	C++ utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx, Trios
Solvers	Iterative linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos, Amesos2
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi, Rbgen
	ILU-type preconditioners	AztecOO, IFPACK, Ifpack2
	Multilevel preconditioners	ML, CLAPS, Muelu
	Block preconditioners	Meros, Teko
	Nonlinear system solvers	NOX, LOCA, Piro
	Optimization (SAND)	MOOCHO, Aristos, TriKota, Globipack, Optipack, ROL
	Stochastic PDEs	Stokhos

Interoperability vs. Dependence

("Can Use") ("Depends On")

- Although most Trilinos packages have no explicit dependence, often packages must interact with *some* other packages:
  - NOX needs operator, vector and linear solver objects.
  - AztecOO needs preconditioner, matrix, operator and vector objects.
  - Interoperability is enabled at configure time.
  - Trilinos **`cmake`** system is vehicle for:
    - Establishing interoperability of Trilinos components…
    - Without compromising individual package autonomy.
    - Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES option
- Architecture supports simultaneous development on many fronts.

A First Program

Solve $Ax = b$

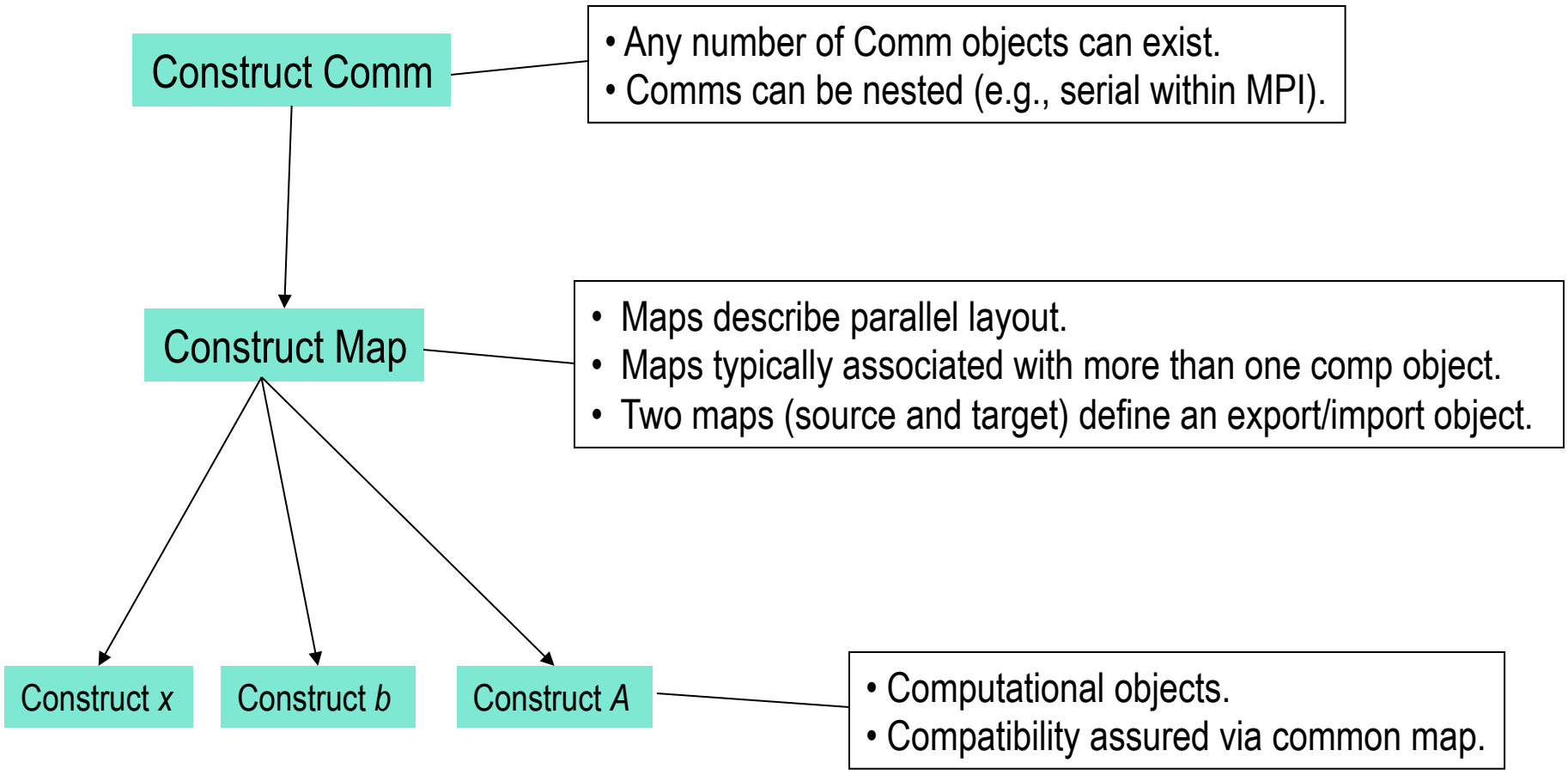
A: Known Matrix

b: Known Right-hand-side.

x: Unknown vector (to be computed).

On a Parallel Computer.

Typical Flow of Epetra Object Construction



A Simple Epetra/AztecOO Program

```
// Header files omitted…
int main(int argc, char *argv[]) {

  Epetra_SerialComm Comm();
```

```
// ***** Map puts same number of equations on each pe *****

  int NumMyElements = 1000 ;
  Epetra_Map Map(-1, NumMyElements, 0, Comm);
  int NumGlobalElements = Map.NumGlobalElements();
```

```
// ***** Create an Epetra_Matrix  tridiag(-1,2,-1) *****

  Epetra_CrsMatrix A(Copy, Map, 3);
  double negOne = -1.0; double posTwo = 2.0;

  for (int i=0; i<NumMyElements; i++) {
    int GlobalRow = A.GRID(i);
    int RowLess1 = GlobalRow - 1;
    int RowPlus1 = GlobalRow + 1;
    if (RowLess1!=-1)
      A.InsertGlobalValues(GlobalRow, 1, &negOne, &RowLess1);
    if (RowPlus1!=NumGlobalElements)
      A.InsertGlobalValues(GlobalRow, 1, &negOne, &RowPlus1);
   A.InsertGlobalValues(GlobalRow, 1, &posTwo, &GlobalRow);
  }
A.FillComplete(); // Transform from GIDs to LIDs
```

```
  // ***** Create x and b vectors *****
  Epetra_Vector x(Map);
  Epetra_Vector b(Map);
  b.Random(); // Fill RHS with random #s
```

```
// ***** Create Linear Problem *****
  Epetra_LinearProblem problem(&A, &x, &b);
```

```
  // ***** Create/define AztecOO instance, solve *****
  AztecOO solver(problem);
  solver.SetAztecOption(AZ_precond, AZ_Jacobi);
  solver.Iterate(1000, 1.0E-8);
```

```
// ***** Report results, finish **********************
  cout << "Solver performed " << solver.NumIters()
       << " iterations." << endl
       << "Norm of true residual = "
       << solver.TrueResidual()
       << endl;


  return 0;
}
```

Sandia National Laboratories

Solver Software Stack

Phase I packages: SPMD, int/double

Phase II packages: Templated

Trilinos

Optimization Unconstrained: Constrained:	Find $u \in \Re^n$ that minimizes $g(u)$ Find $x \in \Re^m$ and $u \in \Re^n$ that minimizes $g(x,u)$ s.t. $f(x,u) = 0$	MOOCHO
Bifurcation Analysis	Given nonlinear operator $F(x,u) \in \Re^{n+m}$ For $F(x,u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$	LOCA
Transient Problems DAEs/ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T], x(0) = x_0, \dot{x}(0) = x_0'$ for $x(t) \in \Re^n, t \in [0,T]$	Rythmos
Nonlinear Problems	Given nonlinear operator $F(x) \in \Re^m \rightarrow \Re$ Solve $F(x) = 0$ $x \in \Re^n$	NOX
Linear Problems Linear Equations: Eigen Problems:	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n$, $\lambda \in$	Anasazi Ifpack, ML, etc... AztecOO
Distributed Linear Algebra Matrix/Graph Equations: Vector Problems:	Compute $y = Ax; A = A(G); A \in \Re^{m \times n}, G \in \Im^{m \times n}$ Compute $y = \alpha x + \beta w; \alpha = \langle x, y \rangle; x, y \in \Re^n$	Epetra Teuchos

Sensitivities (Automatic Differentiation: Sacado)

Solver Software Stack

Phase I packages | Phase II packages | Phase III packages: Manycore*, templated

Trilinos

Optimization Unconstrained: Constrained:	Find $u \in \Re^n$ that minimizes $g(u)$ Find $x \in \Re^m$ and $u \in \Re^n$ that minimizes $g(x,u)$ s.t. $f(x,u) = 0$	Sensitivities (Automatic Differentiation: Sacado)	MOOCHO
Bifurcation Analysis	Given nonlinear operator $F(x,u) \in \Re^{n+m}$ For $F(x,u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$		LOCA | T-LOCA
Transient Problems DAEs/ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T], x(0) = x_0, \dot{x}(0) = x_0'$ for $x(t) \in \Re^n, t \in [0,T]$		Rythmos
Nonlinear Problems	Given nonlinear operator $F(x) \in \Re^m \rightarrow \Re$ Solve $F(x) = 0$ $x \in \Re^n$		NOX | T-NOX
Linear Problems Linear Equations: Eigen Problems:	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n$, $\lambda \in$		Anasazi AztecOO Ifpack, ML, etc... | Belos* Ifpack2*, Muelu*,etc.
Distributed Linear Algebra Matrix/Graph Equations: Vector Problems:	Compute $y = Ax; A = A(G); A \in \Re^{m \times n}, G \in \Im^{m \times n}$ Compute $y = \alpha x + \beta w; \alpha = \langle x, y \rangle; x, y \in \Re^n$		Epetra | Tpetra* Kokkos* Teuchos

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Sandia National Laboratories

Trilinos Provides a Full Collection of Discretization Tools

Shards
definition of cell topology

Intrepid
local (cell-based) FE/FV/FD basis definition; numerical integration; cell geometry; etc.

Phalanx
decomposition of complex PDE systems into a number of elementary user-defined expressions; efficient management of expression dependencies; hooks to embedded tools, etc.

FEI, Panzer
user-defined assignment and management of global degrees of freedom; assembly of local PDE discretization data into distributed linear systems; etc.

Developers: Pavel Bochev, Denis Ridzal, Alan Williams, Roger Pawlowski, Eric, Cyr, others.

Rythmos

- Suite of time integration (discretization) methods
- Includes: backward Euler, forward Euler, explicit Runge-Kutta, and implicit BDF at this time.
- Native support for operator split methods.
- Highly modular.
- Forward sensitivity computations will be included in the first release with adjoint sensitivities coming in near future.

Developers: Curt Ober, Todd Coffey, Roscoe Bartlett

Sandia National Laboratories

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Sandia National Laboratories

Sacado: Automatic Differentiation

- Efficient OO based AD tools optimized for element-level computations

- Applies AD at "element"-level computation
  - "Element" means finite element, finite volume, network device,…
- Template application's element-computation code
  - Developers only need to maintain one templated code base
- Provides three forms of AD
  - Forward Mode: $(x,\ V) \longrightarrow \left(f,\ \frac{\partial f}{\partial x}V\right)$
    - Propagate derivatives of intermediate variables w.r.t. independent variables forward
    - Directional derivatives, tangent vectors, square Jacobians, $\partial f/\partial x$ when **m ≥ n**.
  - Reverse Mode: $(x,\ W) \longrightarrow \left(f,\ W^T\frac{\partial f}{\partial x}\right)$
    - Propagate derivatives of dependent variables w.r.t. intermediate variables backwards
    - Gradients, Jacobian-transpose products (adjoints), $\partial f/\partial x$ when **n > m**.
  - Taylor polynomial mode: $x(t) = \sum_{k=0}^{d} x_k t^k \longrightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}),\quad f_k = \frac{1}{k!}\frac{d^k}{dt^k} f(x(t))$
  - Basic modes combined for higher derivatives.

Developers: Eric Phipps, Carter Edwards (UQ data types)

Sandia National Laboratories

Whirlwind Tour of Packages

Discretizations **Methods** **Core** **Solvers**

Teuchos

- Portable utility package of commonly useful tools:
  - ParameterList class: key/value pair database, recursive capabilities.
  - LAPACK, BLAS wrappers (templated on ordinal and scalar type).
  - Dense matrix and vector classes (compatible with BLAS/LAPACK).
  - FLOP counters, timers.
  - Ordinal, Scalar Traits support: Definition of ‘zero’, ‘one’, etc.
  - Reference counted pointers / arrays, and more…
- Takes advantage of advanced features of C++:
  - Templates
  - Standard Template Library (STL)
- Teuchos::ParameterList:
  - Allows easy control of solver parameters.
  - XML format input/output.

Developers: Chris Baker, Roscoe Barlett, Heidi Thornquist, Mike Heroux, Paul Sexton, Kris Kampshoff, Chris Baker, Mark Hoemmen, many others

Sandia National Laboratories

Trilinos Common Language: Petra

- Petra provides a “common language” for distributed linear algebra objects (operator, matrix, vector)

- Petra[1] provides distributed matrix and vector services.
- Exists in basic form as an object model:
  - Describes basic user and support classes in UML, independent of language/implementation.
  - Describes objects and relationships to build and use matrices, vectors and graphs.
  - Has 2 implementations under development.

[1]**Petra is Greek for “foundation”.**

Petra Implementations

- Epetra (Essential Petra):
  - Current production version.
  - Restricted to real, double precision arithmetic.
  - Uses stable core subset of C++ (circa 2000).
  - Interfaces accessible to C and Fortran users.

- Tpetra (Templated Petra):
  - Next generation C++ version.
  - Templated scalar and ordinal fields.
  - Uses namespaces, and STL: Improved usability/efficiency.
  - Builds on top of Kokkos manycore node library.

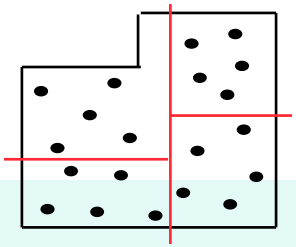
Developers: Chris Baker, Mike Heroux, Rob Hoekstra, Alan Williams, Carter Edwards, Mark Hoemmen, Christian Trott, Siva Rajamanickam

Kokkos: Node-level Data Classes

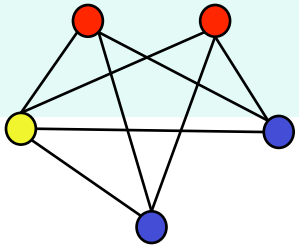
- Manycore/Accelerator data structures & kernels
- Epetra is MPI-only, Tpetra is MPI+[X+Y].
- Kokkos Arrays: Details tomorrow.
  - Simple multi-dimensional arrays.
  - User specifies dimensions and size. Library handles all else.
  - Very good general performance.
- Pretty-good-kernel (PGK) library:
  - Node-level threaded (X) and vector (Y) sparse and dense kernels.
  - Plug replaceable with vendor-optimized libraries.
- Implement Petra Object Model at Node level:
  - Comm, Map/Perm, Vector/Multivector, RowMatrix, Operator.

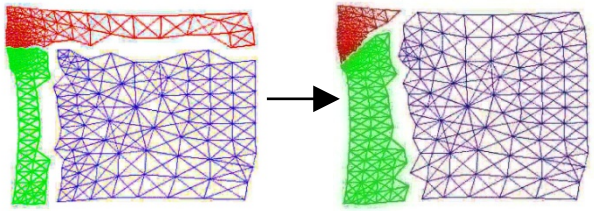
Developer: Mike Heroux, Carter Edwards, Christian Trott, Siva Rajamanickam, etc.

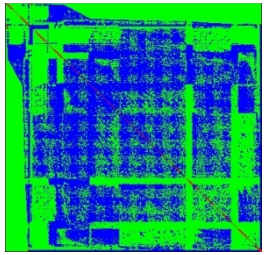
Zoltan/Zoltan2



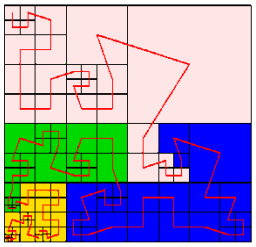
- Data Services for Dynamic Applications
  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering



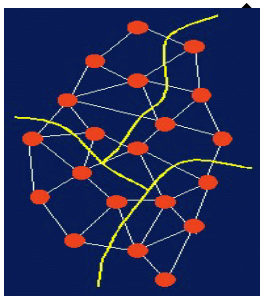




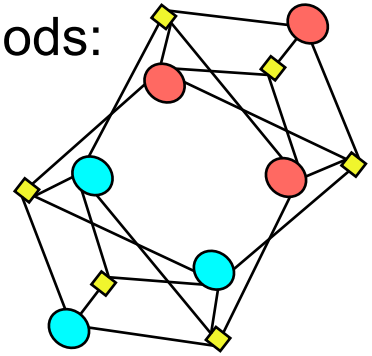
- Partitioners:



Geometric (coordinate-based) methods:
- Recursive Coordinate Bisection (Berger, Bokhari)
- Recursive Inertial Bisection (Taylor, Nour-Omid)
- Space Filling Curves (Peano, Hilbert)
- Refinement-tree Partitioning (Mitchell)



Hypergraph and graph (connectivity-based) methods:
- Hypergraph Repartitioning PaToH (Catalyurek)
- Zoltan Hypergraph Partitioning
- ParMETIS (U. Minnesota)
- Jostle (U. Greenwich)



Developers: Karen Devine, Eric Boman, Siva R., LeAnn Riesen

“Skins”

- PyTrilinos provides Python access to Trilinos packages
- Uses SWIG to generate bindings.
- Epetra, AztecOO, IFPACK, ML, NOX, LOCA, Amesos and NewPackage are supported.

Developer: Bill Spotz

- CTrilinos: C wrapper (mostly to support ForTrilinos).
- ForTrilinos: OO Fortran interfaces.

Developers: Nicole Lemaster, Damian Rouson

- WebTrilinos: Web interface to Trilinos
- Generate test problems or read from file.
- Generate C++ or Python code fragments and click-run.
- Hand modify code fragments and re-run.

Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala, Jim Willenbring

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Sandia National Laboratories

Amesos/Amesos2

- Interface to direct solvers for distributed sparse linear systems (KLU, UMFPACK, SuperLU, MUMPS, ScaLAPACK)

- Challenges:
  - No single solver dominates
  - Different interfaces and data formats, serial and parallel
  - Interface often changes between revisions
- Amesos offers:
  - A single, clear, consistent interface, to various packages
  - Common look-and-feel for all classes
  - Separation from specific solver details
  - Use serial and distributed solvers; Amesos takes care of data redistribution
  - Native solvers: KLU and Paraklete

Developers: Ken Stanley, Marzio Sala, Tim Davis, Siva Rajamanickam

Belos

- Next-generation linear solver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale, linear problems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - **Operator-vector products:** `Belos::MultiVecTraits, Belos::OperatorTraits`
  - **Orthogonalization:** `Belos::OrthoManager, Belos::MatOrthoManager`
  - **Status tests:** `Belos::StatusTest, Belos::StatusTestResNorm`
  - **Iteration kernels:** `Belos::Iteration`
  - **Linear solver managers:** `Belos::SolverManager`

- AztecOO provides solvers for $Ax=b$; what about solvers for:
  - Simultaneously solved systems w/ multiple-RHS: $AX = B$
  - Sequentially solved systems w/ multiple-RHS: $AX_i = B_i$, $i=1,\ldots,t$
  - Sequences of multiple-RHS systems: $A_iX_i = B_i$, $i=1,\ldots,t$

- Many advanced methods for these types of linear systems
  - Block methods: block GMRES [Vital], block CG/BICG [O'Leary]
  - "Seed" solvers: hybrid GMRES [Nachtigal, et al.]
  - Recycling solvers: recycled Krylov methods [Parks, et al.]
  - Restarting techniques, orthogonalization techniques, …

Developers: Heidi Thornquist, Mike Heroux, Mark Hoemmen, Mike Parks, Rich Lehoucq

IFPACK/IFPACK2: Algebraic Preconditioners

- Overlapping Schwarz preconditioners with incomplete factorizations, block relaxations, block direct solves.

- Accept user matrix via abstract matrix interface (Epetra versions).
- Uses Epetra for basic matrix/vector calculations.
- Supports simple perturbation stabilizations and condition estimation.
- Separates graph construction from factorization, improves performance substantially.
- Compatible with AztecOO, ML, Amesos. Can be used by NOX and ML.

Developers: Marzio Sala, Mike Heroux, Siva Rajamanickam, Alan Williams

ML/Muelu: Multilevel Preconditioners

- Smoothed aggregation, multigrid and domain decomposition preconditioning package

- Critical technology for scalable performance of some key apps.
- ML compatible with other Trilinos packages:
  - Accepts user data as Epetra_RowMatrix object (abstract interface). Any implementation of Epetra_RowMatrix works.
  - Implements the Epetra_Operator interface. Allows ML preconditioners to be used with AztecOO, Belos, Anasazi.
- Can also be used completely independent of other Trilinos packages.
- Muelu: Next generation ML (talked about tomorrow).

Developers: Ray Tuminaro, Jeremie Gaidamour, Jonathan Hu, Marzio Sala, Chris Siefert

Anasazi: Eigensolvers

- Next-generation eigensolver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - **Operator-vector products:** `Anasazi::MultiVecTraits`, `Anasazi::OperatorTraits`
  - **Orthogonalization:** `Anasazi::OrthoManager`, `Anasazi::MatOrthoManager`
  - **Status tests:** `Anasazi::StatusTest`, `Anasazi::StatusTestResNorm`
  - **Iteration kernels:** `Anasazi::Eigensolver`
  - **Eigensolver managers:** `Anasazi::SolverManager`
  - **Eigenproblem:** `Anasazi::Eigenproblem`
  - **Sort managers:** `Anasazi::SortManager`
- Currently has solver managers for three eigensolvers:
  - Block Krylov-Schur
  - Block Davidson
  - LOBPCG
- Can solve:
  - standard and generalized eigenproblems
  - Hermitian and non-Hermitian eigenproblems
  - real or complex-valued eigenproblems

Developers: Heidi Thornquist, Mike Heroux, Chris Baker, Rich Lehoucq, Ulrich Hetmaniuk

NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

Broyden's Method

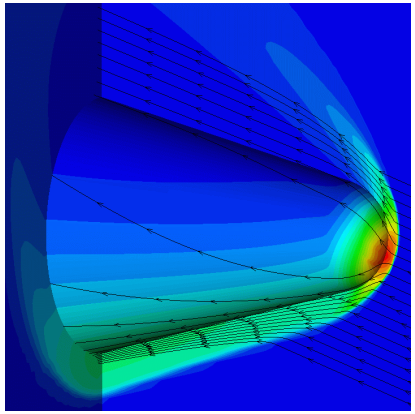
$$M_B = f(x_c) + B_c d$$

Newton's Method

$$M_N = f(x_c) + J_c d$$

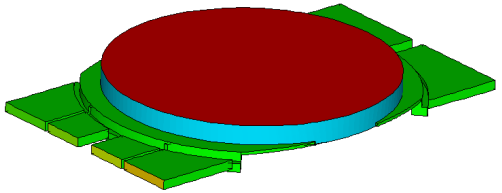
Tensor Method

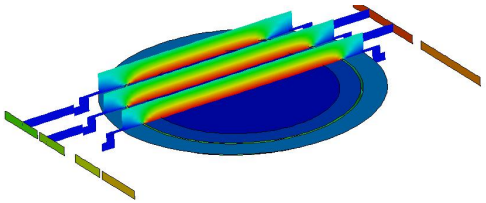
$$M_T = f(x_c) + J_c d + \frac{1}{2} T_c dd$$

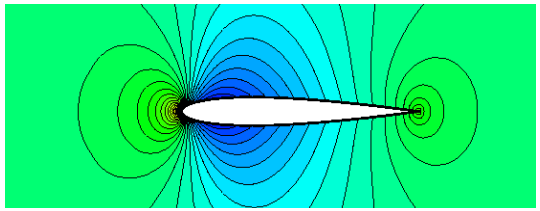


Globalizations

Line Search	Trust Region
Interval Halving	Dogleg
Quadratic	Inexact Dogleg
Cubic	
More'-Thuente	







Jacobian Estimation

- Graph Coloring
- Finite Difference
- Jacobian-Free Newton-Krylov

http://trilinos.sandia.gov/packages/nox

Implementation

- Parallel
- OO-C++
- Independent of the linear algebra package!

Developers: Tammy Kolda, Roger Pawlowski

LOCA

- Library of continuation algorithms

- Provides
  - Zero order continuation
  - First order continuation
  - Arc length continuation
  - Multi-parameter continuation (via Henderson's MF Library)
  - Turning point continuation
  - Pitchfork bifurcation continuation
  - Hopf bifurcation continuation
  - Phase transition continuation
  - Eigenvalue approximation (via ARPACK or Anasazi)

Developers: Andy Salinger, Eric Phipps

Accessing and Controlling Trilinos:
Parameter List and Sublists

```
<ParameterList name="Stratimikos">
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="Amesos">
      <Parameter name="Solver Type" type="string" value="Klu"/>
      <ParameterList name="Amesos Settings">
        <Parameter name="MatrixProperty" type="string" value="general"/>
        ...
        <ParameterList name="Mumps"> ... </ParameterList>
        <ParameterList name="Superludist"> ... </ParameterList>
      </ParameterList>
    </ParameterList>
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-06"/>
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          ...
        </ParameterList>
      </ParameterList>
      ...
    </ParameterList>
    <ParameterList name="Belos"> ... </ParameterList>
  </ParameterList>
<ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
      <ParameterList name="Ifpack Settings">
        <Parameter name="fact: level-of-fill" type="int" value="0"/>
        ...
      </ParameterList>
    </ParameterList>
    <ParameterList name="ML"> ... </ParameterList>
  </ParameterList>
</ParameterList>
```

Top level parameters

Linear Solvers

Preconditioners

Sublists passed on to package code.

Every parameter and sublist is handled by Trilinos code and is fully validated!

Trilinos Integration into an Application

Where to start?

http://trilinos.sandia.gov

Building your app with Trilinos

If you are using Makefiles:

- Makefile.export system

If you are using CMake:

- CMake FIND_PACKAGE

Using CMake to build with Trilinos

- CMake: Cross-platform build system
  - Similar function as the GNU Autotools
- Trilinos uses CMake to build
- You don't have to use CMake to build with Trilinos
- But if you do:
  - FIND_PACKAGE(Trilinos …)
  - Example CMake script in hands-on demo
- I find this much easier than hand-writing Makefiles

Export Makefile System

Once Trilinos is built, how do you link against the application?

There are a number of issues:

- Library link order:
  - -lnoxepetra -lnox –lepetra –lteuchos –lblas –llapack
- Consistent compilers:
  - g++, mpiCC, icc…
- Consistent build options and package defines:
  - g++ -g –O3 –D HAVE_MPI –D _STL_CHECKED

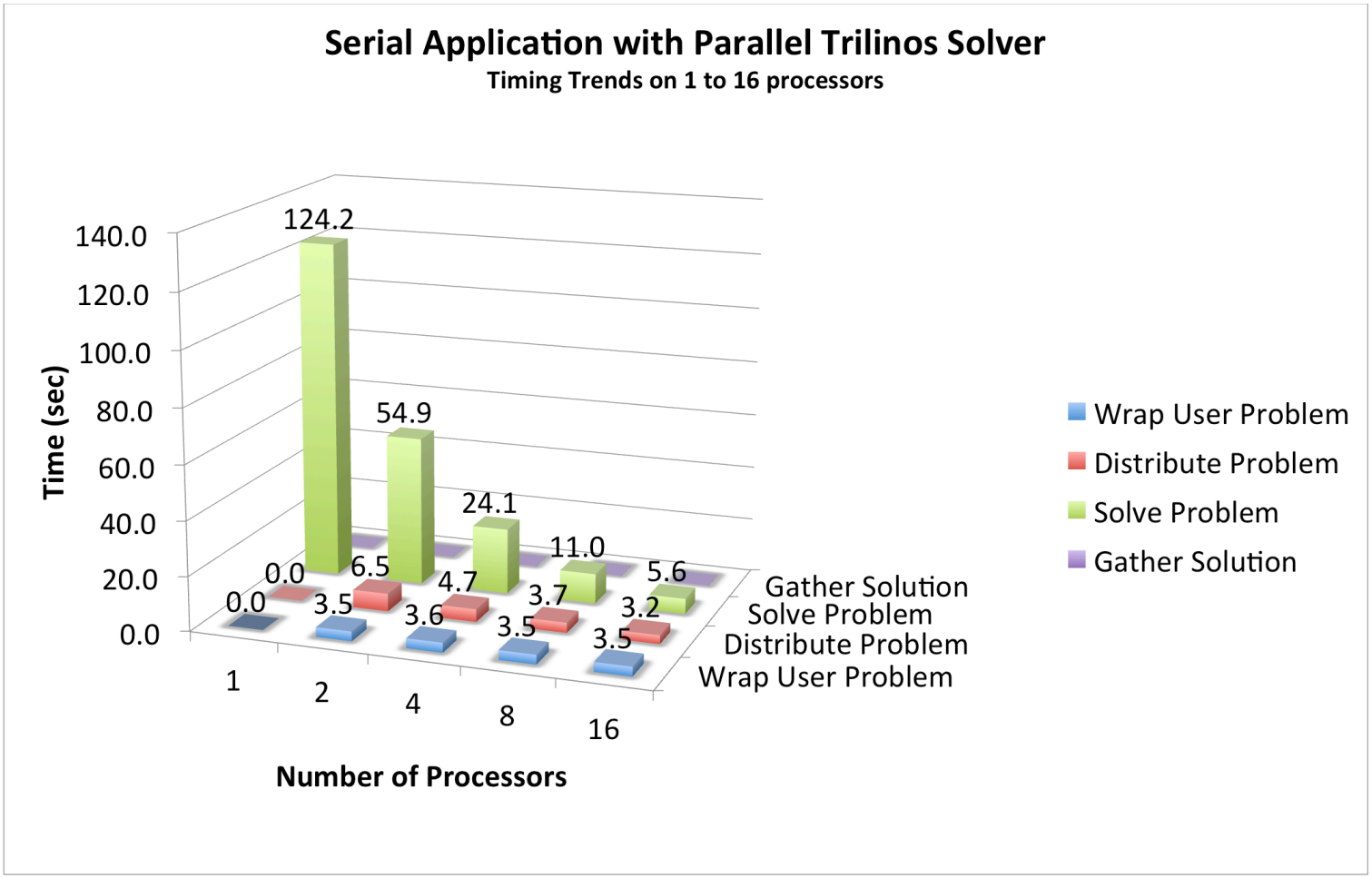
Answer: Export Makefile system

Trilinos Availability / Information

- Trilinos and related packages are available via LGPL or BSD.
- Current release (11.2) is “click release”. Unlimited availability.
- Trilinos Awards:
  - 2004 R&D 100 Award.
  - SC2004 HPC Software Challenge Award.
  - Sandia Team Employee Recognition Award.
  - Lockheed-Martin Nova Award Nominee.
- More information:
  - http://trilinos.sandia.gov (soon trilinos.org).
- Annual Forums:
  - Annual Trilinos User Group Meeting in November @ SNL
    - talks and video available for download
  - Spring Developer Meeting, May @ SNL
  - European meeting (First week in June – Next Year: Lugano, Switzerland).
  - SC’XY Linear Algebra Tutorial (with Dongarra, Demmel, Kurzcak).

First Use of Trilinos: Serial App, Parallel Solver

Small Intel Cluster



Ask For How-to Tech Report if interested.

Trilinos Extreme Scalability

Weak Scaling up to 26.9B Row Matrix

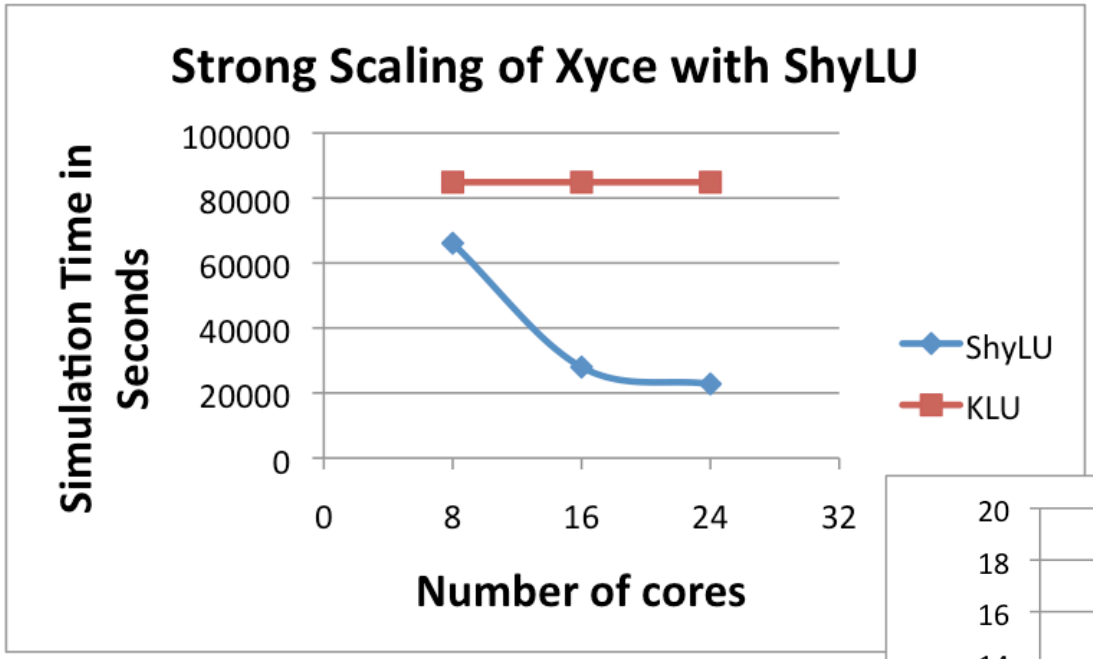
Scaling of CFD to 8.97 billion elements
(coupled momentum matrix with 26.9B rows)
Trilinos/Muelu (Next Generation Preconditioner)

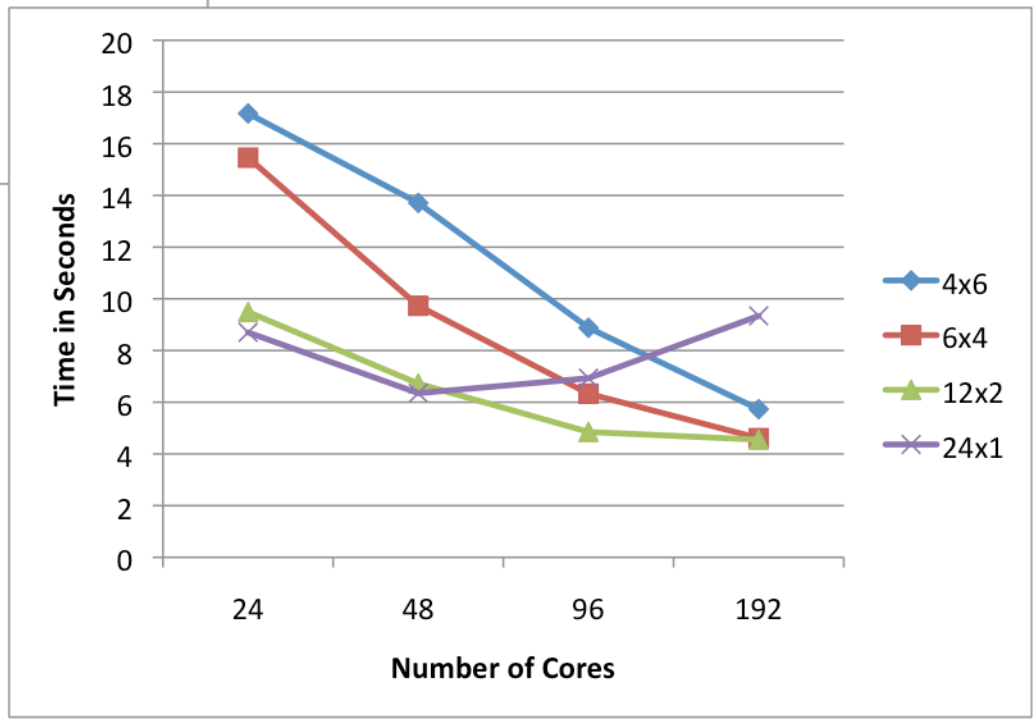
cores	rows	Assem (s)	Assem + Load Complete (s)	Solve (s)	Avg iter
128	52.8M	5.05	5.52	6.1	6.1
1024	421M	5.21	5.76	6.4	6.4
8192	3.37B	5.55	6.35	7.6	7.4
65536	26.9B	5.54	6.71	8.3	8.0

- 100 time steps, 2 nonlinear steps per time step; constant CFL
- 137k elements/core
- Times (sec) and iterations are per nonlinear step

Trilinos for Manycore and Accelerators

Trilinos/ShyLU Manycore Results





S. Rajamanickam, E. G. Boman, and M. A. Heroux. *ShyLU: A hybrid-hybrid solver for multicore platforms*. In Proceedings of 26th International Parallel and Distributed Processing Symposium (IPDPS), Shanghai, China, May 2012.

Intel Value-Added for Trilinos

- Trilinos relies on industry-standard libraries for performance:
  - BLAS/LAPACK: single-node dense linear algebra.
  - MPI: Inter-node communication library.
  - Sparse kernels: Sparse BLAS.
- Intel provides optimized kernels for all of these needs.
- Trilinos is fully compatible with Intel optimized libraries.

Learn More about Trilinos

Hands-On Tutorial

Trilinos website: http://trilinos.sandia.gov

Trilinos Hands-on Tutorial Materials: http://code.google.com/p/trilinos

Trilinos mailing lists: http://trilinos.sandia.gov/mail_lists.html

Trilinos User Group (TUG) meetings (including videos):
http://trilinos.sandia.gov/events/trilinos_user_group_2012
http://trilinos.sandia.gov/events/trilinos_user_group_2011

Summary

- Trilinos provides state-of-the-art solver libraries on state-of-the-art parallel computers.
- Trilinos is the largest single collection of scalable solvers and supporting capabilities in the world.
- Trilinos provides a modern, modular software architecture for reliable, robust parallel computations.
- Trilinos provide portable performance across all modern parallel computers.
- With Intel kernels and communication libraries, Trilinos provides optimal performance across all Intel platforms.